

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-040090

(43)Date of publication of application : 13.02.1998

(51)Int.Cl.

G06F 9/06

(21)Application number : 08-171998

(71)Applicant : INTERNATL BUSINESS MACH  
CORP <IBM>

(22)Date of filing : 02.07.1996

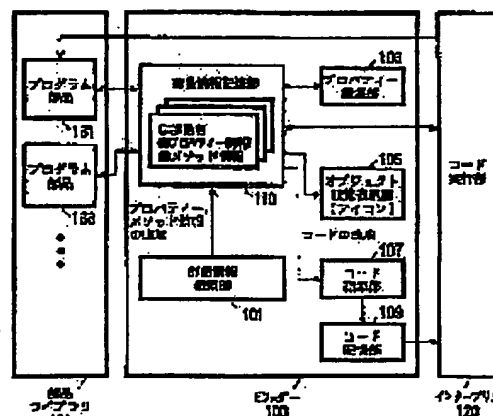
(72)Inventor : HAMADA SEIJI  
NAKAMURA YUICHI  
TAGO KAZUYA

(54) PROGRAM DEVELOPMENT SUPPORTING SYSTEM, ITS METHOD AND STORAGE MEDIUM FOR STORING PROGRAM COMPONENT TO BE USED FOR PROGRAM DEVELOPMENT SUPPORT

(57)Abstract:

PROBLEM TO BE SOLVED: To obtain various programming desired for a programmer by using program components.

SOLUTION: The apparent function of a program component 131 is extended by a component information editing part 101 for updating information stored in a component information storing part 110 for storing information related to the property and method of the program component 131. Component information is used for both of the development and execution of an application program. When the information is used for a property edition part 103 and a code edition part 107 in developing the program, a user can utilize the function of the component for the generation of a programming code as if the function is extended and it is unnecessary to be conscious of a difference between the function of the existing program component and the extended function. When executing the program, the information is used for accessing the component 131 from an interpreter 120 and the extended function is properly executed.



## LEGAL STATUS

[Date of request for examination] 22.12.1999

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 3427918

[Date of registration] 16.05.2003

[Number of appeal against examiner's decision  
of rejection]

[Date of requesting appeal against examiner's  
decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-40090

(43) 公開日 平成10年(1998) 2月13日

(51) Int. Cl.<sup>5</sup>

G 0 6 F 9/08

識別記号

5 3 0

庁内整理番号

F I

G 0 6 F 9/08

技術表示箇所

5 3 0 W

5 3 0 P

審査請求 未請求 請求項の数12 O L (全 28 頁)

(21) 出願番号 特願平8-171998

(22) 出願日 平成8年(1996) 7月2日

(71) 出願人 390009531

インターナショナル・ビジネス・マシー  
ズ・コーポレーション

INTERNATIONAL BUSIN  
ESS MASCHINES CORPO  
RATION

アメリカ合衆国10504、ニューヨーク州

アーモンク (特許なし)

(72) 発明者 横田 誠司

神奈川県大和市下鶴間1623番地14 日本ア

イ・ピー・エム株式会社 大和事業所内

(74) 代理人 弁理士 合田 潔 (外2名)

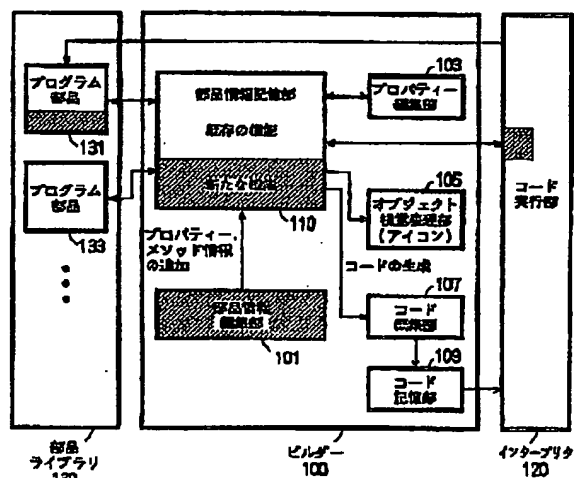
最終頁に続く

(54) 【発明の名称】 プログラム開発支援システム及び支援方法、プログラム開発支援のために用いられるプログラム部品を格納する記憶媒体

(57) 【要約】

【課題】 プログラムが欲する種々多様なプログラミングをプログラム部品を使用して可能にする。

【解決手段】 プログラム部品のプロパティとメソッドに関する情報を保持する部品情報記憶部110の情報を更新する部品情報編集部101により、プログラム部品131の見かけ上の機能を拡張する。部品情報はアプリケーション・プログラム開発時と実行時の両方に利用される。プログラム開発時には、プロパティ編集部103、コード編集部107において利用することにより、ユーザは部品の機能があたかも拡張されたような感覚でプログラミング・コードの作成に利用することができ、しかも既存のプログラム部品の機能と拡張された機能の違いを意識することはない。一方、実行時には、インタプリタ120からプログラム部品131を呼び出すために利用され、拡張された機能を適宜実行することができる。



## 【特許請求の範囲】

【請求項1】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するメソッドとプロパティの情報を提供するプログラム部品を含むデータ処理システムであって、

(a) ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルと、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、

(b) プログラム部品名と、ユーザ定義プロパティ名及びジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルと、ユーザ定義メソッド名及びジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、

(c) 前記プロパティ・テーブルのユーザ定義プロパティ名の登録及び前記メソッド・テーブルのユーザ定義メソッド名の登録を行うための部品編集部と

(d) 前記プログラム部品のユーザ定義プロパティ・バリューを更新するためのプロパティ編集部と、

(e) メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とプロパティ名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、

(f) 前記コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、前記メインロジックを格納するコード記憶部と、

(g) 前記コード記憶部に格納されたメインロジックを受領し、前記メインロジックに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定し、前記プログラム部品に対し、ユーザ定義プロパティ名とジェネリック・プロパティ・アクセス関数を発行し、前記メインロジックに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部と、を含むシステム。

【請求項2】(a) ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルを含むプログラム部品と、

(b) 前記プログラム部品のユーザ定義プロパティ・バリューにアクセスするためのユーザ定義プロパティ名と、ジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルを備える部品情報記憶部と、

(c) 前記プロパティ・テーブルのユーザ定義プロパティ名と前記部品プロパティ・テーブルのユーザ定義プロ

パティ名を登録するための部品編集部とを含むプログラム開発支援システム。

【請求項3】(a) ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、

(b) プログラム部品名とユーザ定義メソッド名と、ジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、

(c) 前記メソッド・テーブルのユーザ定義メソッド名を登録するための部品編集部と

(d) メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、

(e) 前記コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、前記メインロジックを格納するコード記憶部と、

(f) 前記コード記憶部に格納されたメインロジックを受領し、前記メインロジックに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部と、を含むシステム。

【請求項4】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するメソッドとプロパティを提供するプログラム部品と、前記プログラム部品のプロパティ名、プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、実装関数アドレス情報を管理するメソッド・テーブルとを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法であって、

(a) ユーザによるプログラム部品の特定を検出する段階と、

(b) 前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリとメソッド名を入力するためのメソッド情報エントリを表示する段階と、

(c) 前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記プロパティ・テーブルに登録する段階と、

(d) 前記メソッド情報エントリに入力されたユーザ定義メソッド名を前記メソッド・テーブルに登録する段階と、を含む方法。

【請求項5】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法であって、

(a) ユーザによる新規プログラム部品の登録のためのプログラム部品を象徴する新規プログラム部品のアイコンを前記表示装置に表示する段階と、

(b) ユーザによる新規プログラム部品のアイコンの特定を検出する段階と、

(c) 前記特定された新規プログラム部品のプログラム部品名を入力するためのプログラム部品名入力エントリを表示する段階と、

(d) 前記プログラム部品入力エントリに入力されたユーザ定義プログラム部品名に対応したプロパティ名及びプロパティ・バリューのエントリを有する部品プロパティ・テーブルを作成する段階と、

(e) 前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、

(f) 前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記部品プロパティ・テーブルに登録する段階と、

(g) 前記ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、

(h) 前記プロパティ・バリュー入力エントリに入力された値を前記部品プロパティ・テーブルに登録する段階と、を含む方法。

【請求項6】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法であって、

(a) ユーザによるプログラム部品の特定を検出する段階と、

(b) 前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、

(c) 前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記特定されたプログラム部品と関連付ける段階と、

(d) 前記ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、

(e) 前記プロパティ・バリュー入力エントリに入力された値を前記特定されたプログラム部品に関連付ける段階と、を含む方法。

【請求項7】メソッド情報とプロパティ情報を管理するプログラム部品と、前記プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソ

ッド・テーブルと、コード実行部と、プログラム部品名、プロパティ名、メソッド名の記述を含み、前記コード実行部で実行するためのプログラム・コードと前記プログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法であって、

(a) プログラム部品名と前記プログラム部品のプロパティ名と前記プログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、

(b) 前記プログラムコードに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、

(c) 前記プログラム部品名と、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいて前記プログラム部品のプロパティ・バリューを特定する段階と、

(d) 前記プログラムコードに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、

(e) 前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、

(f) プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、

(g) 前記ジェネリック・メソッド・コード・アクセス情報に基づいて、前記コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、

(h) 前記ユーザ定義メソッドのコードを実行する段階と、を含む方法。

【請求項8】プロパティ情報を管理するプログラム部品と、前記プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブルと、コード実行部と、プログラム部品名、プロパティ名の記述を含み、前記コード実行部で実行するためのプログラム・コードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法であって、

(a) プログラム部品名と前記プログラム部品のプロパティ名を指定する記述を含むプログラムコードを受領する段階と、

(b) 前記プログラムコードに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブ

ルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、

(c) 前記プログラム部品名と、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいて前記プログラム部品のプロパティ・バリューを特定する段階と、を含む方法。

【請求項9】メソッド情報を管理するプログラム部品と、前記プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソッド・テーブルと、コード実行部と、プログラム部品名、メソッド名の記述を含み、前記コード実行部で実行するためのプログラム・コードと前記プログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法であって、

(a) プログラム部品名と前記プログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、

(b) 前記プログラムコードに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、

(c) 前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、

(d) プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、

(e) 前記ジェネリック・メソッド・コード・アクセス情報を基づいて、前記コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、

(f) 前記ユーザ定義メソッドのコードを実行する段階と、を含む方法。

【請求項10】実行時において、自ら管理するメソッドとプロパティを提供し、プログラム開発支援のためにプログラム開発支援システムにロードされるプログラム部品を格納する記憶媒体であって、

該プログラム部品は、

(a) ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューのエントリを有する部品プロパティテーブルと、

(b) ユーザ定義メソッド名とジェネリック・メソッド関数を前記プログラム開発支援システムから受けることにより、メソッドのコードをポイントするメソッドコールアダプタと、

を含むことを特徴とする記憶媒体。

【請求項11】入力装置と、表示装置と、コード実行部と、前記コード実行部に自ら管理するメソッドとプロパティを提供するプログラム部品と、前記プログラム部品のプロパティ名、プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、実装関数アドレス情報を管理するメソッド・テーブルとを含むデータ処理システムにおいて、プログラム部品の機能を拡張するためのプログラムを格納する記憶媒体であって、

該プログラムは、

(a) ユーザによるプログラム部品の特定を検出することを前記データ処理システムに指示するプログラムコード手段と、

(b) 前記特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリとメソッド名を入力するためのメソッド情報エントリを表示することを前記データ処理システムに指示するプログラムコード手段と、

(c) 前記プロパティ情報エントリに入力されたユーザ定義プロパティ名を前記プロパティ・テーブルに登録することを前記データ処理システムに指示するプログラムコード手段と、

(d) 前記メソッド情報エントリに入力されたユーザ定義メソッド名を前記メソッド・テーブルに登録することを前記データ処理システムに指示するプログラムコード手段と、を含む記憶媒体。

【請求項12】メソッド情報とプロパティ情報を管理するプログラム部品と、前記プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブル及び、前記プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソッド・テーブルと、コード実行部と、プログラム部品名、プロパティ名、メソッド名の記述を含み、前記コード実行部で実行するためのプログラム・コードと前記プログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、前記コードを編集するためのコード編集部とを含むデータ処理システムにおいて、前記プログラム・コードを実行するためのプログラムを格納する記憶媒体であって、

該プログラムは、

(a) プログラム部品名と前記プログラム部品のプロパティ名と前記プログラム部品のメソッド名を指定する記述を含むプログラムコードを受領することを前記データ処理システムに指示するプログラムコード手段と、

(b) 前記プログラムコードに含まれる部品名とプロパティ名から、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定することを前記データ処理システムに指示するプログラムコード手段と、

(c) 前記プログラム部品名と、前記部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいて前記プログラム部品のプロパティ・バリューを特定することを前記データ処理システムに指示するプログラムコード手段と、

(d) 前記プログラムコードに含まれる部品名とメソッド名から、前記部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定することを前記データ処理システムに指示するプログラムコード手段と、

(e) 前記プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行することを前記データ処理システムに指示するプログラムコード手段と、

(f) プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領することを前記データ処理システムに指示するプログラムコード手段と、

(g) 前記ジェネリック・メソッド・コード・アクセス情報に基づいて、前記コード記憶部に格納されたユーザ定義メソッドのコードにアクセスすることを前記データ処理システムに指示するプログラムコード手段と、

(h) 前記ユーザ定義メソッドのコードを実行することを前記データ処理システムに指示するプログラムコード手段と、  
を含む記憶媒体。

【発明の詳細な説明】

【0001】

【産業上の利用分野】この発明は、プログラムの作成を支援する方法及びシステムに関し、詳しくは、プログラミング構築(開発)支援装置にプログラム部品を統合する方法に関し、さらに詳しくは、プログラム部品が本来持っている機能に加え、新たな機能を追加しながら統合化する方法に関する。

【0002】

【従来の技術】従来より、Visual Basic ("Visual Basic"は、マイクロソフト社の商標)等の対話型プログラム作成支援システムにおいて、プログラマがアプリケーションプログラムを対話的に作成する際、その作成を支援するため、プログラム部品を提供している。図15は、従来のプログラム部品の利用形態を示す図である。

【0003】この例では、ボタン・アイコン203のプログラム部品230が提供されており、プロパティ231として、ボタン・アイコンを表示する位置を定義するための変数情報(x, y)と、ボタン・アイコンの幅と高さ定義するための変数情報(w, h)と、メソッド233として、指定されたx, yの位置にボタン・アイコンを移動することを指示する「move(x, y)」

と、指定されたw, hの大きさにボタン・アイコンを変更する「resize(w, h)」等の情報を含んでいる。

【0004】このプログラム部品がアプリケーション・ウィンドウに位置づけられ、後述するプロパティ編集部でプロパティの値が設定されると、インタプリタはこれらの情報を解釈し、アプリケーション・ウィンドウ上で、設定されたプロパティ・バリューに従ったボタン・アイコンが表示されることとなる。

10 【0005】しかし、このような従来の技術におけるプログラム部品においては、プロパティやメソッドは予め与えられており、利用できるプロパティ、メソッドは限定されたものであった。従って、ボタンの部品に、「ラベルを複数付けたい(従来のプログラミング支援システムにおいて、プログラム部品には、通常、ボタンに1つのラベルしか付すことができない)」、「そのボタンに付されるラベルの文字数に対応させてボタンの大きさを変更したい(従来のプログラミング支援システムにおいて、プログラム部品には、通常、定数値によってサイズを定義し、関数値によってはサイズを定義できない)」といった多種多様なプログラムの要求を満足させることはできず、プログラム部品を利用したプログラミングには多くの制約が存在していた。

30 【0006】また、従来のプログラム部品は、ボタン・アイコンや、ルーラ・バー等の視覚的に図形として表示されるグラフィカル・オブジェクトに対してのみ提供されており、後述するコード編集部で文字や記号を用いて記述される非グラフィカル・オブジェクトに対して提供されていなかった。このため、プログラマは、コーディングを行うプログラミング言語の命令を複数つなぎ合わせ、記述することによりプログラムを作成する必要があった。また、他のプログラムの作成において、かかる記述を再利用することはできず、再度ほぼ同様のコーディングを行う必要があった。サブルーチンを用いたコーディングにおいても、各サブルーチンは、プログラム部品のようなステートを保有できないため、オブジェクト固有の変数の管理が複雑になり、また、その再利用は容易ではない。

40 【0007】かかる従来の問題に関連して、特開平7-28636号公報は、入力操作と視覚オブジェクトを対応付けたテーブルを用いてグラフィカル・ユーザ・インターフェース構築支援装置を提案している。しかし、ここで提案されているグラフィカル・ユーザ・インターフェース構築支援装置は、アプリケーション・プログラム内の利用者による各入力操作に対応する視覚オブジェクトが、その入力操作にどの程度適合しているかを示すため、視覚オブジェクトの候補をその入力操作への適合度の高い順に显示するものであり、プログラム部品の選択を迅速に行うことはできるが、プログラム部品の機能自体を拡張することはできない。

【0008】また、特開平7-219753号公報は、ボタン・アイコンがポインティングデバイスのポインタで押された時等に発生するイベントに対し実行する動作を指示するといったコールバック関数の一部をリソース化することによって、開発効率の向上を行う会話型プログラム開発支援システムを提案している。この方式は、メソッドによって実行される処理内容を変更することができるが、新たなメソッドを追加することはできないため、「そのボタンに付されるラベルの文字数に対応させてボタンの大きさを変更したい」といったプログラムの多種多様な要求を満足させることができない。

【0009】さらに、特開平4-353926号公報は、会話部品定義体入力操作と視覚オブジェクトをテーブルを用いて対応付ける方式を提案している。しかし、この方式は、会話部品に予めメソッドが対応付けられており、そのメソッドをカスタマイズするものであり、会話部品にユーザの欲するメソッドを追加することはできない。このため、「ラベルを複数付けたい」、「そのボタンに付されるラベルの文字数に対応させてボタンの大きさを変更したい」といった多種多様なプログラムの要求を満足させることはできず、また、非グラフィカル・オブジェクトをプログラム部品として定義することができなかった。

【0010】

【発明が解決しようとする課題】この発明の目的は、部品情報記憶部において、プログラム部品が本来持っている機能に関する情報を蓄積するだけでなく、実際の部品にはない新たな機能も蓄積し、プログラムビルダやインタプリタに利用可能にすることにより、従来のプログラム部品では実現できなかったプログラムの種々多様なプログラミングをプログラム部品を利用しつつ実現することである。

【0011】本発明の他の目的は、プログラム部品が元々持っていたプロパティと、新たに追加されたプロパティをプロパティ編集部で区別なく変更可能にすることである。

【0012】本発明の他の目的は、新たに追加されたメソッドやプロパティの利用に際して、プログラム部品が元々持っていたものと区別なくプログラムビルダやインタプリタで利用可能にすることである。

【0013】

【課題を解決するための手段】本発明のアプリケーションプログラム支援システムは、プログラム部品に関する情報を保持する部品情報記憶部を有しており、この情報を更新することにより、プログラム部品の実際の機能を拡張することなく、見かけ上の機能を拡張する。部品情報の更新は部品に新たなプロパティならびにメソッドを追加することに対応しており、このような編集作業は部品情報編集部により行われる。部品情報はアプリケーション・プログラム開発時と実行時の両方に利用される。

すなわち、プログラム開発時には、プロパティ編集部、コード編集部において利用することにより、ユーザは部品の機能があたかも拡張されたような感覚でプログラミング・コードの作成に利用することができ、しかも通常のプログラム部品（ユーザが定義したプログラム部品ではなく、固定的な機能を有するプログラム部品）本来の機能と拡張された機能の違いを意識することはない。一方、実行時には、インタプリタからプログラム部品を呼び出すために利用され、拡張された機能を適宜実行することができる。

【0014】本発明の1の態様においては、ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルと、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、プログラム部品名と、ユーザ定義プロパティ名及びジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルと、ユーザ定義メソッド名及びジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、プロパティ・テーブルのユーザ定義プロパティ名の登録及びメソッド・テーブルのユーザ定義メソッド名の登録を行うための部品編集部とプログラム部品のユーザ定義プロパティ・バリューを更新するためのプロパティ編集部と、メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とプロパティ名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、メインロジックを格納するコード記憶部と、コード記憶部に格納されたメインロジックを受領し、メインロジックに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定し、プログラム部品に対し、ユーザ定義プロパティ名とジェネリック・プロパティ・アクセス関数を発行し、メインロジックに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部とを含むシステムが提供される。

【0015】本発明の他の1態様においては、ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューを管理する部品プロパティテーブルを含むプログラム部品と、プログラム部品のユーザ定義プロパティ・バリューにアクセスするためのユーザ定義プロパティ名と、ジェネリック・プロパティ・アクセス関数を管理するプロパティテーブルを備える部品情報記憶部と、プロパティ・テーブルのユーザ定義プロパティ



名と部品プロパティ・テーブルのユーザ定義プロパティ名を登録するための部品編集部とを含むプログラム開発支援システムが提供される。

【0016】本発明の他の1態様においては、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることにより、ユーザ定義メソッドのコードをポイントするメソッドコールアダプタを含むプログラム部品と、プログラム部品名とユーザ定義メソッド名と、ジェネリック・メソッド関数を管理するメソッド・テーブルを備える部品情報記憶部と、メソッド・テーブルのユーザ定義メソッド名を登録するための部品編集部と、メソッドの実行内容を特定するメソッドのコードを編集するため、および、部品名とメソッド名の指定を含むメインロジックを編集するためのコード編集部と、コード編集部で編集されたメソッドのコードを部品名とメソッド名によって特定可能に格納し、メインロジックを格納するコード記憶部と、コード記憶部に格納されたメインロジックを受領し、メインロジックに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定し、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行するコード実行部とを含むシステムが提供される。

【0017】本発明の他の1態様においては、ユーザによるプログラム部品の特定を検出する段階と、特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリとメソッド名を入力するためのメソッド情報エントリを表示する段階と、プロパティ情報エントリに入力されたユーザ定義プロパティ名をプロパティ・テーブルに登録する段階と、メソッド情報エントリに入力されたユーザ定義メソッド名をメソッド・テーブルに登録する段階と、を含むプログラム部品の機能を拡張する方法が提供される。

【0018】本発明の他の1態様においては、ユーザによる新規プログラム部品の登録のためのプログラム部品を象徴する新規プログラム部品のアイコンを表示装置に表示する段階と、ユーザによる新規プログラム部品のアイコンの特定を検出する段階と、特定された新規プログラム部品のプログラム部品名を入力するためのプログラム部品入力エントリを表示する段階と、プログラム部品入力エントリに入力されたユーザ定義プログラム部品名に対応したプロパティ名及びプロパティ・バリューのエントリを有する部品プロパティ・テーブルを作成する段階と、特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、プロパティ情報エントリに入力された情報を部品プロパティ・テーブルに登録する段階と、ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、プロパティ・バリュー入力エントリに入力された値

を部品プロパティ・テーブルに登録する段階とを含む入力装置と、表示装置と、コード実行部と、コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法が提供される。

【0019】本発明の他の1態様においては、ユーザによるプログラム部品の特定を検出する段階と、特定されたプログラム部品のプロパティ名を入力するためのプロパティ情報エントリを表示する段階と、プロパティ情報エントリに入力されたユーザ定義プロパティ名を特定されたプログラム部品に関連付ける段階と、ユーザ定義プロパティ名に対応するプロパティ・バリューを入力するためのプロパティ・バリュー入力エントリを表示する段階と、プロパティ・バリュー入力エントリに入力された値を特定されたプログラム部品に関連付ける段階とを含む入力装置と、表示装置と、コード実行部と、コード実行部に自ら管理するプロパティ・バリューを提供するプログラム部品とを含むデータ処理システムにおけるプログラム部品の機能を拡張する方法が提供される。

【0020】本発明の他の1態様においては、プログラム部品名とプログラム部品のプロパティ名とプログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、プログラムコードに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、プログラム部品名と、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいてプログラム部品のプロパティ・バリューを特定する段階と、プログラムコードに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、ジェネリック・メソッド・コード・アクセス情報に基づいて、コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、ユーザ定義メソッドのコードを実行する段階とを含むプログラム・コードの実行方法が提供される。

【0021】本発明の他の1態様においては、プログラム部品名とプログラム部品のプロパティ名を指定する記述を含むプログラムコードを受領する段階と、プログラムコードに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定する段階と、プログラム部品名と、部品情報記憶部のプロパティ

・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいてプログラム部品のプロパティ・バリューを特定する段階とを含むプロパティ情報を管理するプログラム部品と、プログラム部品のプロパティ名、ジェネリック・プロパティ・アクセス関数を管理するプロパティ・テーブルと、コード実行部と、プログラム部品名、プロパティ名の記述を含み、コード実行部で実行するためのプログラム・コードを記憶するコード記憶部と、コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法が提供される。

【0022】本発明の他の1態様においては、プログラム部品名とプログラム部品のメソッド名を指定する記述を含むプログラムコードを受領する段階と、プログラムコードに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定する段階と、プログラム部品に対し、ユーザ定義メソッド名とジェネリック・メソッド関数を発行する段階と、プログラム部品が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けること  
20 に応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領する段階と、ジェネリック・メソッド・コード・アクセス情報に基づいて、コード記憶部に格納されたユーザ定義メソッドのコードにアクセスする段階と、ユーザ定義メソッドのコードを実行する段階とを含むメソッド情報を管理するプログラム部品と、プログラム部品のメソッド名、ジェネリック・メソッド関数を管理するメソッド・テーブルと、コード実行部と、プログラム部品名、メソッド名の記述を含み、コード実行部で実行  
30 するためのプログラム・コードとプログラム部品のメソッドに対応付けられて格納されたユーザ定義メソッドのコードを記憶するコード記憶部と、コードを編集するためのコード編集部とを含むデータ処理システムにおけるプログラム・コードの実行方法が提供される。

【0023】本発明の他の1態様においては、ユーザ定義プロパティのユーザ定義プロパティ名、ユーザ定義プロパティ・バリューのエントリを有する部品プロパティテーブルと、ユーザ定義メソッド名とジェネリック・メソッド関数をプログラム開発支援システムから受けることにより、メソッドのコードをポイントするメソッドコールアダプタとを含む、実行時において、自ら管理するメソッドとプロパティを提供し、プログラム開発支援のためにプログラム開発支援システムにロードされるプログラム部品を格納する記憶媒体が提供される。

【0024】本発明の他の1態様においては、ユーザによるプログラム部品の特定を検出することをデータ処理システムに指示するプログラムコード手段と、特定されたプログラム部品のプロパティ名を入力するためのプロ  
50 パティ情報エントリとメソッド名を入力するためのメソ

ッド情報エントリを表示することをデータ処理システムに指示するプログラムコード手段と、プロパティ情報エントリに入力されたユーザ定義プロパティ名をプロパティ・テーブルに登録することをデータ処理システムに指示するプログラムコード手段と、メソッド情報エントリに入力されたユーザ定義メソッド名をメソッド・テーブルに登録することをデータ処理システムに指示するプログラムコード手段とを含むプログラム部品の機能を拡張するためのプログラムを格納する記憶媒体が提供される。

【0025】本発明の他の1態様においては、プログラム部品名とプログラム部品のプロパティ名とプログラム部品のメソッド名を指定する記述を含むプログラムコードを受領することをデータ処理システムに指示するプログラムコード手段と、プログラムコードに含まれる部品名とプロパティ名から、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数を特定することをデータ処理システムに指示するプログラムコード手段と、プログラム部品名と、部品情報記憶部のプロパティ・テーブルで管理されているジェネリック・プロパティ・アクセス関数に基づいてプログラム部品のプロパティ・バリューを特定することをデータ処理システムに指示するプログラムコード手段と、プログラムコードに含まれる部品名とメソッド名から、部品情報記憶部のメソッド・テーブルで管理されているジェネリック・メソッド関数を特定することをデータ処理システムに指示するプログラムコード手段と、プログラム部品  
40 が、ユーザ定義メソッド名とジェネリック・メソッド関数を受けることに応答して発行した、ユーザ定義メソッドのコードをポイントするジェネリック・メソッド・コード・アクセス情報を受領することをデータ処理システムに指示するプログラムコード手段と、ジェネリック・メソッド・コード・アクセス情報に基づいて、コード記憶部に格納されたユーザ定義メソッドのコードにアクセスすることをデータ処理システムに指示するプログラムコード手段と、ユーザ定義メソッドのコードを実行することを前記データ処理システムに指示するプログラムコード手段とを含むプログラム・コードを実行するためのプログラムを格納する記憶媒体が提供される。

【0026】

【実施例】

A. ハードウェア構成

以下、図面を参照して本発明の実施例を説明する。図1を参照すると、本発明を実施するためのハードウェア構成の概観図が示されている。システム100は、中央処理装置(CPU)1とメモリ4とを含んでいる。CPU1とメモリ4は、バス2を介して、補助記憶装置として

のハードディスク装置13とを接続してある。フロッピーディスク装置(またはCD-ROM、MO、MD、ZIP、DVD等の各種記憶媒体の駆動装置)20はフロッピーディスクコントローラ等の各種コントローラ19を介してバス2へ接続されている。

【0027】フロッピーディスク装置(またはCD-ROM等の各種媒体の駆動装置)20には、フロッピーディスク(またはCD-ROM等の各種記憶媒体)が挿入され、このフロッピーディスク等やハードディスク装置13、ROM14には、オペレーティングシステムと協働してCPU等に命令を与え、本発明を実施するためのコンピュータ・プログラムのコードを記録することができ、このコードは、メモリ4にロードされることによって実行される。このコンピュータ・プログラムのコードは圧縮し、または、複数に分割して、複数の媒体に記録することもできる。

【0028】システム100は更に、ユーザ・インターフェース・ハードウェアを備えたシステムとすることができ、ユーザ・インターフェース・ハードウェアとしては、例えば、入力をするためのポインティング・デバイス(マウス、ジョイスティック等)7またはキーボード6や、視覚データをユーザに提示するためのディスプレイ12がある。また、パラレルポート16を介してプリンタを接続することや、シリアルポート15を介してモデムを接続することが可能であり、シリアルポート15およびモデムまたは通信アダプタ18を介して他のコンピュータと通信を行うことが可能である。

【0029】従って、本発明は、通常のパーソナルコンピュータ(PC)、ワークステーションやこれらの組合せによって実施可能であることを容易に理解できるであろう。ただし、これらの構成要素は例示であり、その全ての構成要素が本発明の必須の構成要素となるわけではない。例えば、図1は、スタンド・アロン環境のシステムを示しているが、クライアント/サーバ・システムとして本発明を実現し、サーバ・マシンに本発明のシステム100を配置し、クライアント・マシンは、サーバ・マシンに対して、イーサネット、トークン・リングなどでLAN接続し、クライアント・マシン側には、各種入力のための入力装置と、プログラムの実行結果を見るための表示装置を配置してもよい。また、後述するプログラム部品ライブラリ130または、その一部のプログラム部品をサーバ側に配置し、その他をクライアント側に配置したり、ビルダ100のみをクライアントに配置し、その他をサーバ側に配置する等、設計段階で自由に変更できる。

【0030】オペレーティング・システムとしては、Windows(マイクロソフトの商標)、OS/2(IBMの商標)、AIX(IBMの商標)上のX-WINDOWシステム(MITの商標)などの、標準でGUIマルチウィンドウ環境をサポートするものが望ましい

が、本発明は、後述する非グラフィカルオブジェクトのみをプログラム部品とすることも可能であるため、PC-DOS(IBMの商標)、MS-DOS(マイクロソフトの登録商標)などのキャラクタ・ベース環境でも実現可能であり、特定のオペレーティング・システム環境に限定されるものではない。

#### 【0031】B. システム構成

次に、図2のブロック図を参照して、本発明のシステム構成について説明する。プログラム部品131、133は、部品ライブラリ130に格納されている。このプログラム部品は、その部品のプロパティ及びメソッドに関する情報を管理している。この管理している情報に関しては、後に詳述する。本発明のプログラム開発支援装置(以下“ビルダ”と呼ぶ)はプログラム部品に関する情報を記憶する部品情報記憶部110を含み、その情報は、部品情報編集部101により、プログラマが追加及び修正可能になっている。ビルダ100は、さらに、プロパティ編集部103を有しており、このプロパティ編集部103を使用することによって、プログラム部品131、133のプロパティ・バリューを変更することができる。オブジェクト視覚表現部105は、グラフィカルなプログラム部品を表示する。

【0032】また、プログラマは、コード編集部107を使用してプログラムのコーディングを行うことができ、コーディングされたプログラム・コードは、コード記憶部109に格納される。そして、インタプリタ120は、このコード記憶部109に格納されたプログラム・コードと、プログラム部品の情報に基づいてプログラムを実行する。本発明の好適な実施例においては、コード実行部120は、インタプリタにより実施されている。しかし、コード実行部120はインタプリタには限定される概念ではなく、コンパイラ等によっても実施可能である。なお、このブロック図は、発明の理解の容易化を図るため一定の機能を有する構成を1つのまとまりとして表現したものであり、例えば、部品情報編集部101とプロパティ編集部103を統合した部品情報編集部として実施したり、また、部品情報編集部101を、部品プロパティ情報編集部と部品メソッド情報編集部に分割して実施することもできる。

【0033】本発明と従来技術との差異を明確化するために、図3において、本発明のシステム構成の一実施例における従来技術との差異部分(斜線部分)を示す。この図において、プログラム部品131は、本発明のプログラム部品であり、プログラム部品133は、従来のプログラム部品である。本発明のビルダは、従来のプログラム部品と本発明のプログラム部品を区別することなく取り扱うことができることを示している。また、部品情報編集部101が新たに追加され、部品情報記憶部110、インタプリタ120に変更が加えられていることが示されている。

# 【0034】B1. 部品情報記憶部プロパティ・テーブル

図4は、本発明の好適な実施例における、部品情報記憶部のプロパティ・テーブルを示す図である。この部品情報記憶部のプロパティ・テーブル（以下単に「プロパティ・テーブル」と呼び、後述する部品プロパティ・テーブルと区別する）は、プログラム部品の部品名に対応して存在する。プロパティ・テーブル150には図に示すように、プロパティ名155、プロパティタイプ156、Get関数157、Put関数158及びユーザ定義フラグ159を管理している。

【0035】このGet関数157、Put関数158は、プログラム部品131、133にアクセスし、プログラム部品131、133からそのプロパティの値をGetし、または、プログラム部品131、133へ、プロパティの値をPutするための関数である。このGet関数157、Put関数158は、プログラム部品131、133の性質によって、Get関数157のみ必要であり、Put関数158は不要である場合（そのプログラム部品からデータをもらうだけ）や、Put関数158のみ必要であり、Get関数157は不要である場合（そのプログラム部品へデータを書き出すだけ）も存在する。本願明細書において、このGet関数157、Put関数158の総称として「プロパティ・アクセス関数」を用いる。

【0036】プロパティ・テーブル150は、既存のプロパティ情報部分151と、ユーザ定義のプロパティ情報部分153に分かれている。既存のプロパティ情報部分151は、プログラム部品133がシステムにインストールされると、そのプログラム部品133の有しているプロパティ名、プロパティ・タイプ、Get関数、Put関数の情報がこのプロパティ・テーブル150にコピーされ登録される。

【0037】この一方、ユーザ定義のプロパティ情報部分153は、部品情報編集部101でプログラマがプログラム部品を指定して、追加の登録を行うと、登録されたプロパティ名、プロパティ・タイプがセットされ、プロパティ・アクセス関数（Get関数、Put関数）が自動生成される。本明細書において、ユーザがプロパティ名をまたはメソッド名を指定して、定義したプロパティまたはメソッドを「ユーザ定義プロパティ」または「ユーザ定義メソッド」と呼び、その他従来のプロパティやメソッドを「既存のプロパティ」、「既存のメソッド」と呼ぶこととする。

【0038】このプロパティ・アクセス関数は、本願発明の好適な実施例においては、「GetProperty/PutProperty」という、ユーザ定義プロパティ共通のプロパティ・アクセス関数がセットされる（以下、この複数のユーザ定義のプロパティ情報に共通のプロパティ・アクセス関数を「ジェネリック・プロパ

ティ・アクセス関数」と呼ぶ）。

【0039】図に示すように、既存のプロパティ情報部分151と、ユーザ定義のプロパティ情報部分153が共通の形式で管理されているため、プロパティ編集部103、プログラムビルダ100、及びインタープリタ120は、既存のプロパティ情報部分151と同様にユーザ定義のプロパティ情報部分153を扱うことができる。但し、既存のプロパティの情報と、ユーザ定義のプロパティの情報を共通の形式で管理する必要は必ずしも必須ではなく、部品情報記憶部110のプロパティ・テーブル150にアクセスするプロパティ編集部103等に変更を加え、2種類のテーブルにアクセス可能にすることにより、本発明を実施することもできる。さらに、既存のプロパティの情報と、ユーザ定義のプロパティの情報をテーブルで管理する必要は必ずしもなく、プロパティ名をキーに、それに対応した情報にアクセス可能な形式で、情報が格納されていれば、本発明を実施することができる。

【0040】本発明の好適な実施例においては、このユーザ登録により作成されたプロパティには、ユーザが定義したプロパティであることを示すフラグ159がセットされる。但し、かかるフラグは本発明の必須の構成要素ではなくシステムが既存のプロパティ情報であるか、ユーザ定義のプロパティ情報であるかを判別できればよい。かかる判別の手法としては、既存のプロパティ情報と、ユーザ定義のプロパティ情報の格納する位置を変え、その位置情報で区別する方法や、ユーザ定義のプロパティ情報のプロパティ名を既存のプロパティ情報のプロパティ名と区別できる名前（例えばプロパティ名の先頭または最後を「USR」にする）によって判別することも可能である。

【0041】B2. 部品情報記憶部メソッド・テーブル  
図5は、本発明の好適な実施例における、部品情報記憶部のメソッド・テーブルを示す図である。この部品情報記憶部のメソッド・テーブル（以下単に「メソッド・テーブル」と呼ぶ）は、プロパティ・テーブル150と同様にプログラム部品の部品名に対応して存在し、既存のメソッド情報161と、ユーザ登録のメソッド情報163によって構成されている。メソッド・テーブル160の既存のメソッド情報161には図に示すように、メソッド名165、実装関数アドレス情報（本発明の好適な実施例においてはC言語が使用されている）167、メソッド・タイプ168、及びユーザ定義フラグ169を管理している。

【0042】この実装関数アドレス情報167は、このメソッド・テーブルが管理しているプログラム部品において、そのメソッドが実行された場合に起動されるモジュールをポインタするためのアドレス情報を管理している。例えば、そのプログラム部品が、ボタン・アイコンの部品であり、メソッドが「Move」であった場合、そ

のボタン・アイコンを移動するために実行されるプログラムモジュールをポイントするためのアドレス情報を管理している。この既存のメソッド情報の実装関数は、プログラム部品が管理しているため、実装関数アドレス情報は、プログラム部品131、133の実装関数をポイントするためのアドレスである。メソッド・タイプ168は、そのメソッドの実行に必要なプロパティを定義している。具体的には、「引き数1、引き数2、...、引き数n/戻り値1、戻り値2、...、戻り値m」の形式で定義されている。メソッドは、その種類によって、引き数のみを有するもの、戻り値のみを有するもの、双方を有するもの、双方とも有していないものが存在する。

【0043】ユーザ定義のメソッド情報163も同様に、メソッド名165、実装関数アドレス情報167、メソッド・タイプ168、及びユーザ定義フラグ169を管理している。このように、既存のメソッド情報部分161と、ユーザ定義のメソッド情報部分163が共通の形式で管理されているため、プロパティ編集部103、プログラムビルダ100、及びインタープリタ120は、既存のメソッド情報部分161と同様にユーザ定義のメソッド情報部分163を扱うことができる。ユーザ定義のメソッド情報163において、この実装関数アドレス情報167は、1つのプログラム部品に対して定義された、複数のユーザ定義のメソッド情報に共通のものであり、後述するプログラム部品131のメソッド・コール・アダプタ137をポイントするためのアドレス情報を管理している。

【0044】プログラム部品におけるプロパティ情報と同様に、既存のメソッド情報部分161は、プログラム部品133がシステムにインストールされると、そのプログラム部品133の有しているメソッド名、実装関数、メソッド・タイプの情報がこのメソッド・テーブル160にコピーされ登録される。この一方、ユーザ定義のメソッド情報部分163は、部品情報編集部101でプログラマがプログラム部品を指定して、追加の登録を行うと、登録されたメソッド名、メソッド・タイプがセットされ、実装関数が自動生成される。

【0045】この実装関数は、本発明の好適な実施例においては、「Call Method」という、ユーザ定義メソッド共通のプログラム部品に存在する見せかけの実装関数をポイントするための情報がセットされる（以下、この複数のユーザ定義のメソッド情報に共通の実装関数アドレス情報を「ジェネリック・メソッド関数」と呼ぶ）。また、本発明の好適な実施例においては、このユーザ登録により作成されたメソッドには、ユーザが定義したプロパティと同様に、ユーザが定義したメソッドであることを示すフラグ169がセットされる。但し、かかるフラグも、プロパティ情報におけるフラグ159と同様に、本発明の必須の構成要素ではなくシステムが

既存のメソッド情報であるか、ユーザ定義のメソッド情報であるかを判別できればよい。また、プロパティ情報において述べたように、既存のメソッド情報とユーザ定義のメソッド情報は、共通の形式で管理することは必須ではない。また、テーブルの形式で管理することも必須ではない。

【0046】B3. プログラム部品プロパティ情報

次に、プログラム部品131で管理されている情報について説明する。プログラム部品131においても、部品情報記憶部110で管理されている情報と同様に、プログラム部品のプロパティに関する情報と、プログラム部品のメソッド情報に関する情報を管理している。ただし、部品情報管理部110において説明したように、プログラム部品は、その性質上、プロパティ情報のみ、またはメソッド情報のみを管理している部品も存在するため、この2種類の情報を同時に有していることは本願発明の必須の構成要件とはならない。

【0047】図6は、本発明の好適な実施例における、プログラム部品のプロパティ情報（以下「部品プロパティ情報」という）およびプログラム部品のプロパティ・テーブル（以下部品情報記憶部11のプロパティ・テーブルと区別するため、「部品プロパティ・テーブル」と呼ぶ）を示す図である。部品プロパティ情報171で管理されている情報は、部品情報記憶部110において説明した既存のプロパティ情報151に対応する情報であり、部品プロパティ・テーブル173で管理されている情報は、ユーザ定義のプロパティ情報153に対応する情報である。部品プロパティ情報171は、その名が示すように、本発明の好適な実施例においても、テーブルの形式で管理されていない。プログラム部品がシステム導入されるときに、プログラム部品は、この部品プロパティ情報171の内容をビルダ100側に宣言するため、ビルダ100は、その情報の内容や種類を認識することができる。

【0048】部品プロパティ情報171は、部品情報管理部110の既存のプロパティにおけるプロパティ・テーブル151の情報とほぼ同様の情報を管理している。プロパティ・テーブル151と部品プロパティ情報171の管理する情報で異なる点は、部品プロパティ情報171においては、プロパティに対する実際のプロパティ・バリューの情報を管理していることである。このプロパティ・バリュー179は、プログラム部品131のインストール時にデフォルトでセットされている場合や、プロパティ編集部103によってその値が設定される場合とがある。

【0049】この一方、部品プロパティ・テーブル173においては、プロパティ名175、タイプ177、およびプロパティ・バリュー179の情報が管理されている。このプロパティ・テーブル173の情報は、部品情報編集部101におけるプログラマの登録によってセッ

トされる。なお、プロパティ・バリュー179の情報は、プロパティ編集部103から入力することもできる。

#### 【0050】B4. プログラム部品メソッド情報

次に、メソッドに関する情報がプログラム部品131においてどのように管理されているかを説明する。図7は、本発明の好適な実施例における、プログラム部品のメソッド情報（以下単に「部品メソッド情報」と呼ぶ）を示す図である。部品メソッド情報180は、部品情報記憶部110におけるメソッド・テーブル160と同様に既存のメソッド情報のメソッド名185、メソッド・タイプ188を管理している。実装関数187は、部品情報記憶部110におけるメソッド・テーブル160と異なり、そのメソッドが実行された場合に起動されるモジュールを管理している。

【0051】ユーザ定義のメソッド情報に関しては、プログラム部品131には、既存のメソッド情報のようにプログラム部品側で管理されておらず、この代替として、メソッド・コール・アダプタ137が存在する。メソッド・コール・アダプタの具体的な内容は後述するが、ユーザ定義のメソッド情報のメソッドが実行された場合に起動されるモジュールをポイントするための関数が管理されている。

#### 【0052】C. システムの動作

本発明の好適な実施例においては、プログラム部品の機能を拡張するために、図3で示したように、部品情報記憶部110、プログラム部品131、インタプリタ120をそれぞれ拡張し、新たに部品情報編集部101を追加している。以下では、部品情報のユーザ定義、ビルダーによる部品情報の利用、実行時における部品情報の利用、という観点から各部分の拡張およびシステムの動作について説明する。

#### 【0053】C1. 部品情報のユーザ定義

図8は、本発明の好適な実施例における、部品情報をプログラマ（ユーザ）が定義する際のデータの流れを示す図である。部品情報の更新、すなわち、プロパティの追加、変更、削除、及び、メソッドの追加、変更、削除は、部品情報編集部101が提供する機能によって行うことができる。

#### 【0054】(1). プロパティの定義

プログラマは、部品情報編集部101において、プログラム部品の指定を行った後、プロパティ情報を入力するためのエントリを表示させることができる。本発明の好適な実施例においては、プログラム部品は、新規プログラム部品登録用アイコン、または既存のプログラム部品アイコンとして提供されているため、そのプログラム部品のアイコンをクリックすることにより選択することができる。

【0055】プログラマが指定したプログラム部品がすでに存在している場合（本発明の好適な実施例において

は、既に登録されているプログラム部品はプルダウンメニューにて選択可能になっている）は、そのプログラム部品のプロパティ情報、がユーザに修正及び削除可能に表示される。プログラマは、プロパティ・タイプ156を修正することができ、また、プロパティ名155を修正すると、修正後の新たなプロパティとして登録される。プログラマが選択したプログラム部品が新規なものである場合、プログラマは、プロパティ名155、プロパティ・タイプ156を入力する。本発明の好適な実施例においては、この更新（追加／修正／削除）に際し、シンタックスチェック等、内部的に矛盾の生ずる更新をチェックし、矛盾の生ずる更新を自動訂正またはリジェクトすることができる。

【0056】このユーザによる登録操作により既存のプロパティ情報の更新の場合、内部的には部品情報記憶部110内のプロパティ・テーブル153のプロパティ情報が更新され（既存のプロパティ情報の更新の場合、プロパティ・タイプ156が更新され、既存のプロパティの削除の場合、そのプロパティのプロパティ名155、プロパティ・タイプ156、プロパティ・アクセス関数157、158、フラグ159が削除される）、さらに対応するプログラム部品131内の部品プロパティ情報171の情報もこれに対応して変更される。

【0057】プログラマが指定したプログラム部品がシステムに存在していない新規なものである場合は、ユーザの指定したプログラム部品名が部品情報記憶部110と、対応するプログラム部品に登録され、登録されたプログラム部品のプロパティ・テーブル153にユーザが定義したプロパティ名155、プロパティ・タイプ156が登録され、プロパティ・アクセス関数157、158にジェネリック・プロパティ・アクセス関数が自動的にセットされ、フラグ159にユーザ定義プロパティであることを判別するための“1”が自動的にセットされる。そして対応するプログラム部品131内の部品プロパティ・テーブル173のプロパティ名175、プロパティ・タイプ177にユーザの定義したプロパティ名、プロパティ・タイプにセットされる。

#### 【0058】(2). メソッドの定義

プロパティの定義と同様に、プログラマは、部品情報編集部101において、プログラム部品の指定を行うと、メソッド情報を入力するためのエントリが表示される。プログラマが指定したプログラム部品がすでに存在している場合は、そのプログラム部品のメソッド情報がユーザに修正及び削除可能に表示される。プログラマは、メソッド・タイプ168を修正することができ、また、メソッド名165を修正すると、修正後の新たなメソッドとして登録される。プログラマが選択したプログラム部品が新規なものである場合、プログラマは、メソッド名165、メソッド・タイプ168を入力する。本発明の好適な実施例においては、この更新（追加／修正／削

除)に際し、シンタックスチェック等、内部的に矛盾の生ずる更新をチェックし、矛盾の生ずる更新を自動訂正またはリジェクトすることができる。

【0059】このユーザによる登録操作により、既存のメソッド情報の更新の場合、内部的には部品情報記憶部110内のメソッド・テーブル163のメソッド情報が更新され(既存のメソッド情報の更新の場合、メソッド・タイプ168が更新され、既存のメソッドの削除の場合、そのメソッドのメソッド名165、実装関数アドレス情報167、メソッド・タイプ168フラグ169が削除される)、さらに対応するプログラム部品131内の部品メソッド情報180の情報もこれに対応して変更される。

【0060】プログラマが指定したプログラム部品がシステムに存在していない新規なものである場合は、ユーザの指定したプログラム部品名が部品情報記憶部110と、対応するプログラム部品に登録され、登録されたプログラム部品のメソッド・テーブル163にユーザが定義したメソッド名165、メソッド・タイプ168が部品情報編集部101で定義された内容にセットされ、実装関数アドレス情報167にジェネリック・メソッド関数が自動的にセットされ、フラグ159にユーザ定義メソッドであることを判別するための“1”が自動的にセットされる。この点において、部品情報編集部101と、部品情報記憶部110の動きは、プロパティ情報の場合と同様であるが、メソッド情報の場合は、本発明の好適な実施例においては、部品情報記憶部110内のメソッド・テーブル163のメソッド情報のみが更新され、プログラム部品131に対しての更新はない。これは、後述するメソッド・コール・アダプタの働きによって、他の場所に格納されているメソッドのコードを利用できるからである。但し、プログラム部品において、十分な記憶域が確保できるのであれば、プロパティの場合と同様に、プログラム部品側でメソッドのコードを管理することも可能である。

【0061】C2. ユーザ定義部品情報によるプログラム作成支援

次に、このようにユーザによって定義されたプログラム部品がプログラム作成においてどのように利用されるかを説明する。図9は、本発明の好適な実施例における、ビルダがプログラム部品情報を利用する際のデータの流れを示す図である。

【0062】(1). プロパティ・バリューの設定  
プログラマは、プロパティ編集部103を用いて、指定したプログラム部品のプロパティ・バリューを設定することができる。本発明の好適な実施例においては、部品情報記憶部110のプロパティ・テーブル150は、既存のプロパティとユーザ定義のプロパティを共通のテーブルで管理しているため、プロパティ編集部103は、従来のプロパティ編集部の機能に変更を加えなくても、

部品情報記憶部110のユーザ定義のプロパティ・テーブル153のプロパティ情報にアクセスすることができる。但し、部品情報記憶部110のプロパティ・テーブル150は、既存のプロパティ・テーブル151とユーザ定義のプロパティ・テーブル153を共通の形式で管理することは必須ではなく、プロパティ編集部103を、別々で異なる既存のプロパティ・テーブル151とユーザ定義のプロパティ・テーブル153の2つのテーブルにアクセスできるようにプロパティ編集部の機能を拡張する事によって実施することができる。

【0063】プログラマが定義したプロパティ・バリューが既存のプロパティ情報であった場合、ビルダ100は、プロパティ・テーブル151のプロパティ・アクセス関数157、158を使用し、従来技術による手法を用いて、プログラム部品131の部品プロパティ情報171のプロパティ・バリュー179にアクセスし、プロパティ編集部103を用いて、指定したプログラム部品のプロパティ・バリューを設定することができる。

【0064】プログラマが指定したプロパティ・バリューがユーザ定義プロパティのプロパティ・バリューであった場合、ビルダ100は、部品情報記憶部110のプロパティ・テーブル153に格納されているプロパティ名155とジェネリック・プロパティ・アクセス関数157、158を用いて、プログラム部品131の部品プロパティ・テーブル173のプロパティ・バリュー179にアクセスし、プロパティ編集部103を用いて、プロパティ・バリューを設定することができる。

【0065】具体的に説明すると、ビルダ100は、まず、ユーザ定義のプロパティ・バリューをプロパティ編集部103で登録可能にするために、部品情報記憶部110のプロパティ・テーブル150に格納されているプロパティ名155を表示する。また、これに対応するプロパティ・バリューを表示するために、まず、そのプロパティが既存のプロパティであるか、ユーザ定義のプロパティであるかをプロパティ・テーブル150のフラグ159を検査することによって判断する。

【0066】ユーザ定義のプロパティであると判断された場合は、ジェネリック・プロパティ・アクセス関数のうち、“GetProperty”157を用いて、プログラム部品131の部品プロパティ・テーブル173にアクセスする。そして、部品プロパティ・テーブル173のプロパティ・バリュー179をプロパティ名175をキーに検索し、その値をプロパティ編集部103に渡すことによって、プロパティ・バリュー179をプロパティ編集部103に表示可能にする(登録された値がない場合は、空白が表示される)。

【0067】そして、プログラマが、プロパティ編集部103のプロパティ・バリューを変更すると、ビルダ100は、部品情報記憶部110のプロパティ・テーブル153に格納されているプロパティ名155とジェネリ

ック・プロパティ・アクセス関数のうち、"Put Property"158を用いて、プログラム部品131の部品プロパティ・テーブル173のプロパティ・バリュー179にアクセスし、プロパティ・バリューを更新する。

#### 【0068】(2)．メソッド情報の利用

プログラムはコード編集部107において、ユーザ定義プログラム部品のメソッドを特定する記載をすることができ、これにより、コード実行部120において実行することができる、詳しくは後に詳述する。

【0069】C3. ユーザ定義部品情報を使用して作成されたプログラムの実行

次に、ユーザによって定義されたプログラム部品を使用して作成されたプログラムがどのように実行されるかを説明する。

【0070】図10は、本発明の好適な実施例における、プログラム実行時でのプログラム部品情報を利用する際のデータの流れを示す図である。まず、既存（ユーザ定義の情報でなく、従来のプログラム部品の情報）のプロパティ情報、メソッド情報の呼出については、インタプリタ120がコード記憶部109に記述されたコードを解釈し、ここで、プログラム部品名とプロパティ名、または、プログラム部品名とメソッド名が指定されていると、インタプリタ120は、プログラム部品の情報を利用することができる。

#### 【0071】(1) プロパティ情報の利用

図11は、本発明の好適な実施例における、プログラム実行時でのユーザ定義プロパティ情報を利用する際のデータの流れを示す図である。インタプリタ120がプログラム部品のプロパティ情報を利用する場合、インタプリタは、ビルダ100に対し、プログラム部品131のプロパティ・バリュー179にアクセスするための情報を求める。これに対し、ビルダ100は、部品情報記憶部110のプロパティ・テーブル150をインタプリタ120から渡されたプロパティ名をキーに検索する。ビルダ100は、インタプリタ120が要求するプロパティが、ユーザ定義のプロパティ情報が既存のプロパティ情報であるかを、部品情報記憶部110のプロパティ・テーブル150のフラグ159を検査することにより判断する。

【0072】既存のプロパティ情報であった場合は、ビルダ100は、インタプリタ120に、プロパティ・テーブル150のプロパティ・アクセス関数157、158を返す。このとき、インタプリタ120は、コードの内容から、プログラム部品のプロパティ・バリューへのアクセスがGetであるかPutであるかを判断することができるため、インタプリタ120は、ビルダ100に対し、Getプロパティ・アクセス関数157、Putプロパティ・アクセス関数158のいずれかを指定して受領する方式と、Getプロパティ・アクセス関数1

57と、Putプロパティ・アクセス関数158の両方を受領し、そのいずれか一方を選択する方式とがある。

【0073】インタプリタ120は、この情報によって、プログラム部品131、133の部品プロパティ情報にアクセスし、プロパティ・バリュー179にアクセスし、Getプロパティ・アクセス関数157の場合は、プロパティ・バリュー179の読み取りを行い、Putプロパティ・アクセス関数158の場合は、プロパティ・バリュー179への書き込みを行う。

【0074】この一方、ユーザ定義のプロパティである場合は、ビルダ100は、インタプリタ120に、プロパティ名を引き数としたジェネリックプロパティ・アクセス関数157、158を返す。このときも既存のプロパティ情報へのアクセスと同様に、インタプリタ120は、コードの内容から、プログラム部品のプロパティ・バリューへのアクセスがGetであるかPutであるかを判断することができるため、インタプリタ120は、ビルダ100に対し、Getジェネリック・プロパティ・アクセス関数157、Putジェネリック・プロパティ・アクセス関数158のいずれかを指定して受領する方式と、Getジェネリック・プロパティ・アクセス関数157と、Putジェネリック・プロパティ・アクセス関数158の両方を受領し、そのいずれか一方を選択する方式とがある。

【0075】また、インタプリタ120が、ビルダ100に対し、プロパティ名をキーに、プロパティがユーザ定義プロパティであるか、既存のプロパティであるかの情報をもとめ、ビルダ100は、プロパティ・テーブル150のフラグ159のみをインタプリタ120に返し、インタプリタ120側でジェネリック・プロパティ・アクセス関数を発行してもよい。さらに、プログラム部品131へのプロパティ情報登録時に、プログラム部品名、プロパティ名をインタプリタ120に渡すことにより、インタプリタ120は、ビルダ100側にユーザ定義のプロパティか、既存のプロパティかの情報の問い合わせを行う必要はなくなる。

【0076】インタプリタ120は、この情報によって、プログラム部品131の部品プロパティ・テーブルにアクセスし、また、プロパティ・バリュー179にアクセスし、Getジェネリック・プロパティ・アクセス関数157の場合は、プロパティ・バリュー179の読み取りを行い、Putジェネリック・プロパティ・アクセス関数158の場合は、プロパティ・バリュー179への書き込みを行う。

#### 【0077】(2) メソッド情報へのアクセス

図12は、本発明の好適な実施例における、プログラム実行時でのユーザ定義メソッド情報を利用する際のデータの流れを示す図である。既存のメソッドコールは部品のメソッドに対応する関数を直接呼び出すことにより行われる。しかしながら、新たに追加されたメソッドに対



応する関数は、プログラム部品側に存在せず、インタプリタ120が管理している。そこで、グミの関数を用意し、インタプリタからこの関数を関数名とともに呼び出すことにする。インタプリタは、プロパティ情報とほぼ同様な方式に従って、メソッド名を引き数としたジェネリック・メソッド関数を発行する。

【0078】プログラム部品131のメソッド・コール・アダプタ137は、インタプリタ120から受領したジェネリック・メソッド関数を、インタプリタ120で管理するメソッドのコード123を格納する位置をポイントする情報（以下「ジェネリック・メソッド・コード・アクセス情報」と呼ぶ）に変換する。本発明の好適な実施例においては、ジェネリック・メソッド・コード・アクセス情報は、メソッド名を引き数として、インタプリタ側に返される。本発明の好適な実施例においては、メソッドのコード123は、コード編集部107によって作成され、メソッド名をキーに、インタプリタ120が検索可能な状態でコード記憶部109に格納されており、インタプリタ120は、ジェネリック・メソッド・コード・アクセス情報を受領すると、メソッド名をキーにメソッドのコード123を検索し、メソッドのコード123を実行する。

【0079】D. プログラム作業手順

次に、本願発明の好適な実施例におけるプログラム部品の利用形態について、プログラムの作業手順に沿って説明する

【0080】D1. 部品情報定義手順

図15乃至19は、本発明の好適な実施例における、プログラム部品の設定手順を示す図であり、図15は、プログラム部品の設定手順における初期状態を示す図である。図において、メインウィンドウ310は、ビルダのメインウィンドウであり、アプリケーション・ウィンドウ302に対応する各種フォームや、コード記憶部109に格納された各種コードを管理するためのウィンドウである。

【0081】アプリケーション・ウィンドウ320は、図2、3に示すオブジェクト視覚表現部105に対応するウィンドウである。初期状態において、何も情報が入っておらず、プログラマはこのアプリケーション・ウィンドウ320を用いてプログラム部品の指定等を行う。このアプリケーション・ウィンドウによって、プログラマは直感的に理解しやすいプログラミングを行うことが可能になる。但し、後述する部品ライブラリの表示機能を拡張することによってもプログラム部品の指定を示すことが可能であるため、このアプリケーション・ウィンドウは本発明の必須の構成要素ではない。

【0082】図の左上に配置された複数のアイコンの集合体は部品ライブラリ330である。このアイコン1つ1つがプログラム部品に対応している。これらのアイコンのうち、UDObjと記されたアイコン331はユー

ザ定義プログラム部品専用に提供されているアイコンである。プログラマが新たなプログラム部品を作成する場合、図16に示すように、まず、アプリケーション・ウィンドウ320に、UDObjアイコン331をドロップする（アイコン321）。アプリケーション・ウィンドウ320にアイコン331を位置づける手法は、ドラッグドロップの他、アイコン331をダブルクリックしたり、アイコン331を選択した状態において、予め設定されたキーをおす等、種々存在し、設計段階でさまざまに設定できる事項である。

【0083】そして、図17に示すように、プログラマがメイン・ウィンドウ310のウィンドウの項目をクリックし、プルダウンを表示させ、プロパティ・エディタの項目を選択すると、プロパティ・エディタ・ウィンドウ340がオープンする。このアイコンに対応したウィンドウをオープンさせる手法や手順は、例えば、アプリケーション・ウィンドウ320に位置づけられたアイコン321をダブルクリックするとプロパティ・エディタ・ウィンドウ340がオープンする等種々存在し、当業者は設計段階でそれらの手法や手順から所望のものを選択し実施化することができる。本発明の好適な実施例においては、新規登録用のアイコン331がアプリケーション・ウィンドウ320にドロップされると、プロパティ・エディタ・ウィンドウ340が自動的にオープンし、カーソルはプログラム部品名入力エントリ347に位置づけられ、強調表示によってプログラマにプログラム部品名の入力を促す。なお、このプロパティ・エディタ・ウィンドウ340は図2、3のプロパティ編集部103に対応するウィンドウである。

【0084】プログラマは、まず、プログラム部品名の設定を行うことができる。図において、プログラマは、プロパティ・エディタ・ウィンドウ340のプログラム部品名入力エントリ347にキーボード入力することにより、プログラム部品名を、“Order”と設定している。そして、登録と記されたボタン・アイコン341をクリックすることによって、プログラマの入力の内容を登録する。この操作により、プログラム部品名“Order”に対応して、部品情報記憶部110に空のプロパティ・テーブル150、メソッド・テーブル160が作成され、また、部品ライブラリ130に、プログラム部品名“Order”の新たなプログラム部品が作成される。

【0085】なお、本発明の好適な実施例においては、この初期登録状態において、部品情報記憶部110に登録されるプロパティ・テーブル150等は、空ではなく、どのアプリケーション・ウィンドウに位置づけられたかを示す“Container”や、インデックス等のプロパティ名や、プロパティ・バリューが自動的にセットされ、部品情報記憶部110のプロパティ・テーブル150や、プログラム部品のプロパティ情報に登録され

る。このプログラム部品を部品ライブラリ330にアイコンとして表示し、選択可能な状態にすることも可能である。なおプロパティ・エディタ・ウィンドウ340には、キャンセルのボタン・アイコン343が提供されており、このボタンがクリックされるとプロパティ・エディタ・ウィンドウ340の各エントリに入力された未登録の文字列が無効にされる。

【0086】プログラム部品名の設定が終わると、次にプログラマは、プロパティやメソッドの登録を行う。図18において表示されているウィンドウ350は、オブジェクト・ウィンドウであり、図2、3における部品情報編集部に対応している。オブジェクト・ウィンドウ350にはプログラム部品名355が表示されているため、プログラマは、プログラム部品"Order"のプロパティまたはメソッドを定義していることを容易に知ることができる。図18において、まず、プロパティの登録が行われている。プロパティの定義は、プロパティ入力エントリ357に、プロパティ名とプロパティ・タイプをキーボード入力することにより行うことができる。また、一覧と記されたボタン・アイコン361をクリックすると、定形的なプロパティ名とプロパティ・タイプがプルダウンメニューの形式で表示され、プログラマは、その一覧の中から1つを選択し、利用することができる。

【0087】このプロパティ入力エントリ357に記載された文字列は、登録のボタン・アイコン363がクリックされることにより登録される。プロパティの登録がなされると、オブジェクト・ウィンドウ350においては、登録されているプロパティのリスト359に登録された定義の内容が追加される。図においては、"ProductID as long"というプロパティの定義文が登録されたことが示されている。

【0088】このプロパティの登録がなされると、部品情報記憶部110のプロパティ・テーブル150のプロパティ名155、プロパティ・タイプが、ユーザの定義した内容に従って更新され、また、プロパティ・アクセス関数157、158のエントリにジェネリック・プロパティ・アクセス関数を示す情報がセットされる。また、フラグ159に、ユーザ定義のプロパティであることを示す「1」がセットされる。同様に、プログラム部品131においても、部品プロパティ・テーブル173のプロパティ名とプロパティ・タイプが登録される。なお、プロパティの登録のボタン・アイコンがクリックされたことを検出して、プロパティ・エディタ・ウィンドウ340を登録後の内容で再表示させることも可能である。

【0089】図19において、プロパティの登録に次いでメソッドの登録が行われている。プロパティの登録とメソッドの登録は、どちらを先に行ってもよいが、本明細書においては、読者の負担軽減のため、プロパティを

先に、メソッドを後に説明を行うこととする。メソッドの登録の、プロパティの登録と同様に、入力エントリ365に、メソッド名と、メソッド・タイプをキーボード入力することにより行うことができる。また、一覧と記されたボタン・アイコン369をクリックすると、定形的なメソッド名とメソッド・タイプがプルダウンメニューの形式で表示され、プログラマは、その一覧の中から1つを選択し、利用することができる。

【0090】このメソッド入力エントリ365に記載された文字列は、登録のボタン・アイコン363がクリックされることにより登録される。メソッドの登録がなされると、オブジェクト・ウィンドウ350においては、登録されているメソッドのリスト367に登録された定義の内容が追加される。図においては、"QueryPrice (ID as long) as string"というメソッドの定義文が登録されたことが示されている。

【0091】このメソッドの登録がなされると、部品情報記憶部110のメソッド・テーブル160のメソッド名165、メソッド・タイプ168が、ユーザの定義した内容に従って更新され、また、実装関数アドレス情報167のエントリにジェネリック・メソッド関数を示す情報が自動的にセットされる。また、フラグ169に、ユーザ定義のメソッドであることを示す「1」がセットされる。

#### 【0092】D2. プログラム作成手順

次に、上述のユーザ定義のプログラム部品の情報が、プログラム開発時に、どのようにプログラマに用いられるかを説明する。図20、21は、本発明の好適な実施例における、プログラム部品の利用手順を示す図である。

図20において、プログラマは、ユーザ定義プロパティのプロパティ・バリューの設定を行っている。プログラマは、プロパティ・エディタ・ウィンドウ340を再表示させると、ユーザ定義プロパティに対応したエントリが提供されているため、プロパティ・バリューを設定することができる。

【0093】これは、プロパティ・エディタ・ウィンドウ340の再表示が行われると、ビルダ100は、部品情報記憶部110のプロパティ・テーブル150に格納されているプロパティ名155をすべて表示する。また、これに対応するプロパティ・バリューを表示するために、まず、そのプロパティが既存のプロパティであるか、ユーザ定義のプロパティであるかをプロパティ・テーブル150のフラグ159を検査する事によって判断する。ユーザ定義のプロパティである場合は、フラグ159の値が1であるため判断することができる。この場合、ジェネリック・プロパティ・アクセス関数のうち、"GetProperty"157を用いて、プログラム部品131の部品プロパティ・テーブル173にアクセスする。そして、部品プロパティ・テーブル173

のプロパティ・バリュー179をプロパティ名175をキーに検索し、その値をプロパティ編集部103に渡すことによって、プロパティ・バリュー179をプロパティ編集部103に表示する。この段階において、ユーザ定義のプロパティのプロパティ・バリューは何も設定されていない状態なので、空白が表示されることとなる。

【0094】そして、プログラマが、プロパティ名“Product”に対し、プロパティ・バリュー“PS/55”、“(「PS/55」はIBM社の商標)、プロパティ名“ProductID”に対し、プロパティ・バリュー“5433”を入力すると、ビルダ100は、部品情報記憶部110のプロパティ・テーブル153に格納されているプロパティ名155とジェネリック・プロパティ・アクセス関数のうち、“PutProperty”158を用いて、プログラム部品131の部品プロパティ・テーブル173のプロパティ・バリュー179にアクセスし、プロパティ・バリューを更新する。

【0095】次に、コード編集部におけるプログラム開発時に、どのようにプログラマに用いられるかを説明する。図21において、プログラマは、プログラムのコーディングをしている。プログラマは、このコード・エディタ・ウィンドウ380において、ユーザ定義メソッドのコードの作成と、メインロジック部のコード作成を行うことができる。図において、ウィンドウ380はコード・エディタ・ウィンドウであり、図2、3におけるコード編集部に対応している。この図において、プログラム部品名“Order”、メソッド名“QueryCustomer”のユーザ定義メソッドのコードを定義している。図におけるコード記述エントリ393で記載されたコードの内容が、コード記憶部109に格納され、インタプリタ120に渡されることにより、ユーザ定義メソッドのコードとして実行されることとなる。ここに記載されているメソッドのコードの記述から、他のプログラム部品のプロパティやメソッドを使用するための記述と同様のコーディングで、メインロジックの記述を行うことができる。コード記述エントリ393には、図に示すように、他のプログラム部品の部品名395とプロパティ397を指定し、または、他のプログラム部品の部品名399とメソッド名401を引き数等を付加して指定することにより他のプログラム部品のプロパティやメソッドを利用することもできる。

【0096】このコード・エディタのウィンドウでは、モジュール・エントリ387、オブジェクト・エントリ389、メソッド・エントリ391の3種類のエントリが提供されている。プログラマはこれらのエントリに、直接キーボード入力することによって、コード記述エントリに入力したコードの部品名とメソッド名を指定する。各エントリ387、389、391には、プルダウンメニューの機能を有している。

【0097】モジュール・エントリ387のプルダウン

・メニューは、他のフォーム（コード・エディタで編集した他のコード）を呼び出すために設けられている。このプルダウン・メニューで選択されたフォームが、コード記述エントリ393に表示される。オブジェクト・エントリ389のプルダウン・メニューは、他のプログラム部品を検索し選択するためのプルダウン・メニューである。メソッド・エントリ391のプルダウン・メニューは、他のメソッドを検索し選択するためのプルダウン・メニューである。

【0098】実行のためのボタン・アイコン381がクリックされると、このコードがインタプリタ120に渡され、前述の手順に従ってコードが実行される。なお、実行のためのボタン・アイコン381の他に中断、中止のためのボタン・アイコン383、385が提供されており、このアイコンをクリックすることにより、プログラムの実行を中断または中止することができる。

【0099】

【発明の効果】以上説明したように、本発明によれば、部品情報記憶部において、プログラム部品が本来持っている機能に関する情報を蓄積するだけでなく、実際の部品にはない新たな機能も蓄積し、プログラム構築支援構築装置内の他の装置やインタプリタが利用できるようにする。

【0100】また、本発明によれば、部品情報編集部において、プログラム部品に新たなプロパティならびにメソッドを追加することにより部品のみかけ上の機能を拡張することができる。

【0101】また、部品情報記憶部に保存された情報を基にして、プログラム部品が元々持っていたプロパティと、新たに追加されたプロパティをプロパティ編集部で区別なく変更することができる。

【0102】そして、部品情報記憶部に保存された情報に基づいて、メソッドの骨格のコードを自動的に生成し、新たに追加されたメソッドをコード編集部で記述し、新たに追加されたメソッドやプロパティの利用に際して、部品が元々持っていたものと区別なく利用することができる。

【0103】さらに、プログラムの実行に関して、インタプリタからプログラム部品を呼び出す場合に、部品情報記憶部に保存された情報を用いて、新たに追加されたプロパティの参照・変更、ならびにメソッドを呼び出すことができる。

【0104】プログラム部品は、通常C言語等の言語によって記述されているため、従来は、あるプログラム部品に対して新たにプロパティやメソッドを追加することは、これと異なる言語であるベーシック言語等によって行うことができず、C言語等によって行わなければならなかった。しかし、本発明のプログラム部品へのプロパティ情報やメソッド情報の追加方法を用いれば、ベーシック言語等のよりユーザにとって理解しやすい言語で

コーディングすることが可能となり、開発効率も向上することができる。

【0105】

【図面の簡単な説明】

【図1】 ハードウェア構成の一実施例を示すブロック図である。

【図2】 本発明のシステム構成の一実施例を示す機能ブロック図である。

【図3】 本発明のシステム構成の一実施例における従来技術との差異部分を示す機能ブロック図である。

【図4】 本発明の好適な実施例における、部品情報記憶部のプロパティ・テーブルを示す図である。

【図5】 本発明の好適な実施例における、部品情報記憶部のメソッド・テーブルを示す図である。

【図6】 本発明の好適な実施例における、プログラム部品のプロパティ・テーブルを示す図である。

【図7】 本発明の好適な実施例における、プログラム部品のメソッド・テーブルを示す図である。

【図8】 本発明の好適な実施例における、部品情報を更新する際のデータの流れを示す図である。

【図9】 本発明の好適な実施例における、ビルダが部品情報を利用する際のデータの流れを示す図である。

【図10】 本発明の好適な実施例における、プログラム実行時での部品情報を利用する際のデータの流れを示す図である。

【図11】 本発明の好適な実施例における、プログラム実行時での部品情報を利用する際のデータの流れを示す図である。

【図12】 本発明の好適な実施例における、プログラム実行時での部品情報を利用する際のデータの流れを示す図である。

【図13】 従来のプログラム部品の利用例を示す図である。

【図14】 従来のプログラム作成支援システムを示す\*

\* 図である。

【図15】 本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図16】 本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図17】 本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図18】 本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

10 【図19】 本発明の好適な実施例における、プログラム部品の設定手順を示す図である。

【図20】 本発明の好適な実施例における、プログラム部品の利用手順を示す図である。

【図21】 本発明の好適な実施例における、プログラム部品の利用手順を示す図である。

【0106】

【符号の説明】

100 ビルダ (プログラム開発支援装置)

101 部品情報編集部

103 プロパティ編集部

105 オブジェクト視覚表現部

107 コード編集部

109 コード記憶部

110 部品情報記憶部

120 インタプリタ

130 部品ライブラリ

131, 133 プログラム部品

137 メソッドアダプタ

150 プロパティ・テーブル

160 メソッド・テーブル

171 部品プロパティ情報

173 部品プロパティ・テーブル

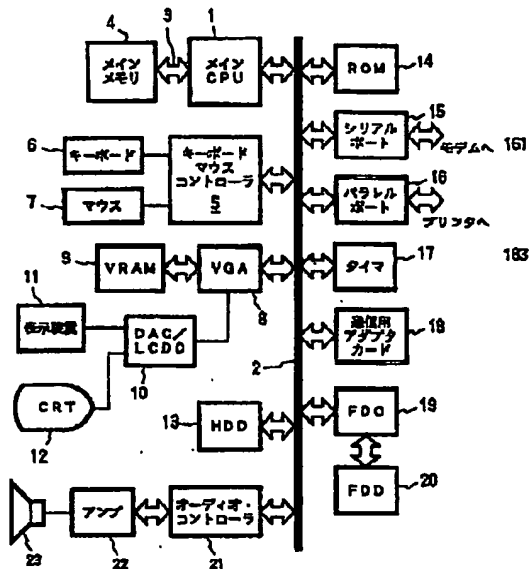
180 部品メソッド情報

【図4】

名前 155	タイプ 156	Get関数 157	Put関数 158	フラグ 159	
Width	Integer	GetWidth	PutWidth	0	(通常の プロパティ)
Label	String	GetLabel	PutLabel	0	
⋮	⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	⋮	
Aaa	Integer	GetProperty	PutProperty	1	(ユーザ定義の プロパティ)
Bbb	String	GetProperty	PutProperty	1	
⋮	⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	⋮	

プロパティ・テーブル 159

【図 1】

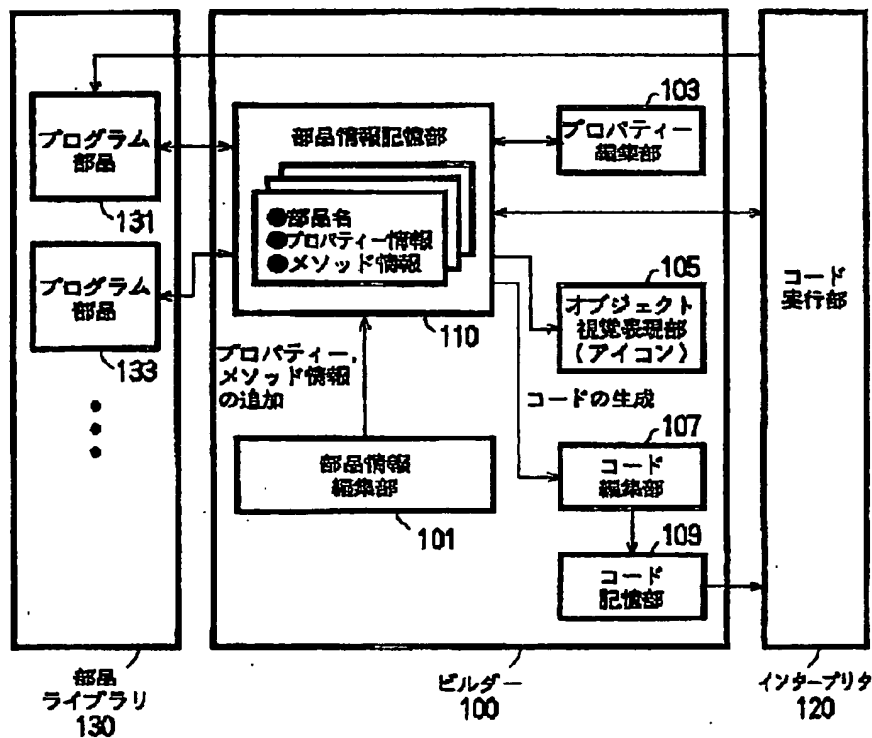


【図5】

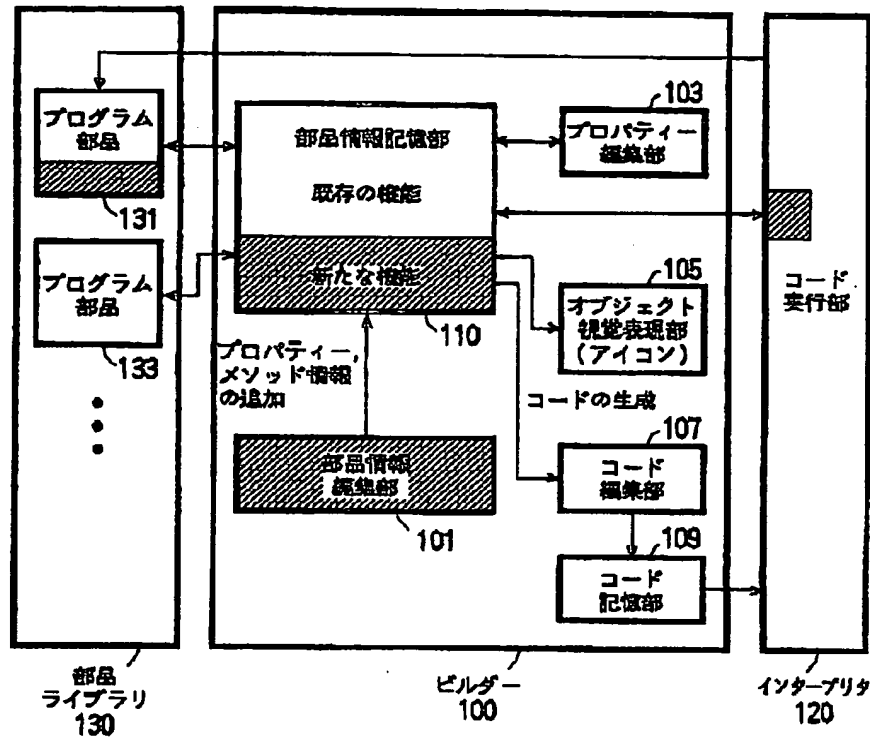
名前 185	実装アドレス情報 187	タイプ 188	フラグ 189
Move Rename GetStatus : :	Moveアドレス情報 Renameアドレス情報 GetStatusアドレス情報 : :	x as Integer, y as Integer/ a as String/~ ~/Return Code as Integer : :	0 0 0 : :
Xxx Yyy : :	CallMethodアドレス情報 CallMethodアドレス情報 : :	a as Integer a as String : :	1 1 : :

メソッド・テーブル 160

【图2】



【図3】



【図6】

名前 175	タイプ 176	GETメソッド 177	PUTメソッド 178	値 179
Width	Integer	GetWidth	PutWidth	値
Label	String	GetLabel	PutLabel	値
⋮	⋮	⋮	⋮	⋮

部品プロパティ情報 171

【図7】

名前 185	実装メソッド 187	タイプ 188
Move	Moveアドレス情報	x as Integer, y as Integer/~
Rename	Renameアドレス情報	a as String/~
GetStatus	GetStatusアドレス情報	-/ReturnCode as Integer
⋮	⋮	⋮

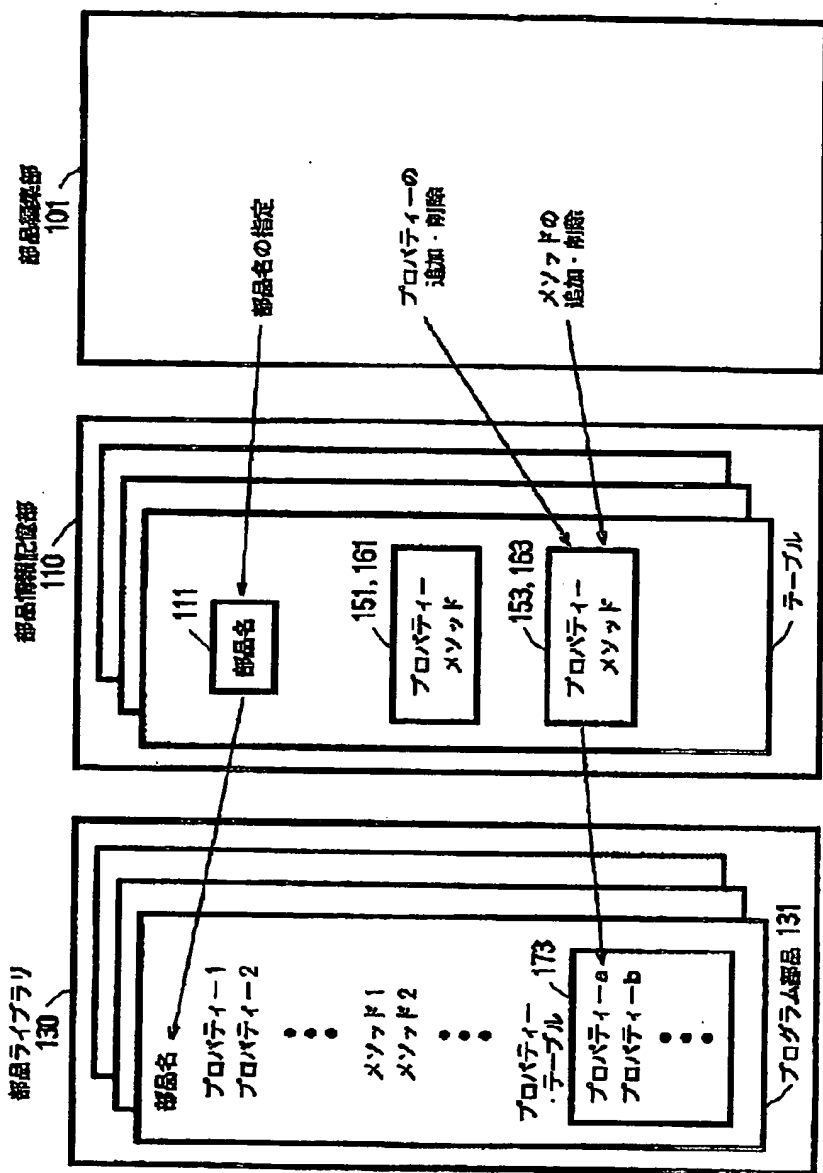
部品メソッド情報 180

名前 175	タイプ 177	値 179
Aaa	Integer	値
Bbb	String	値
⋮	⋮	⋮

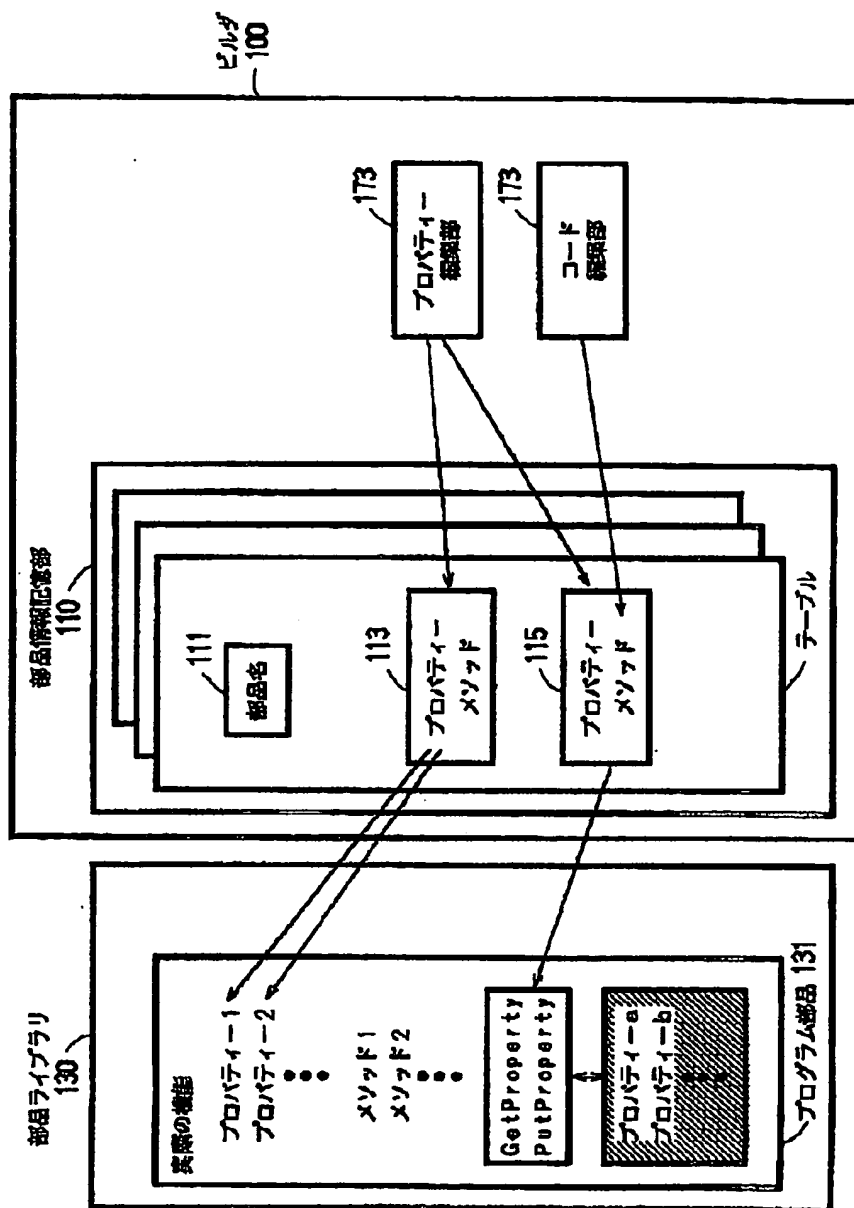
部品プロパティ・テーブル 173

メソッド・コール・アダプタ  
181

【図8】

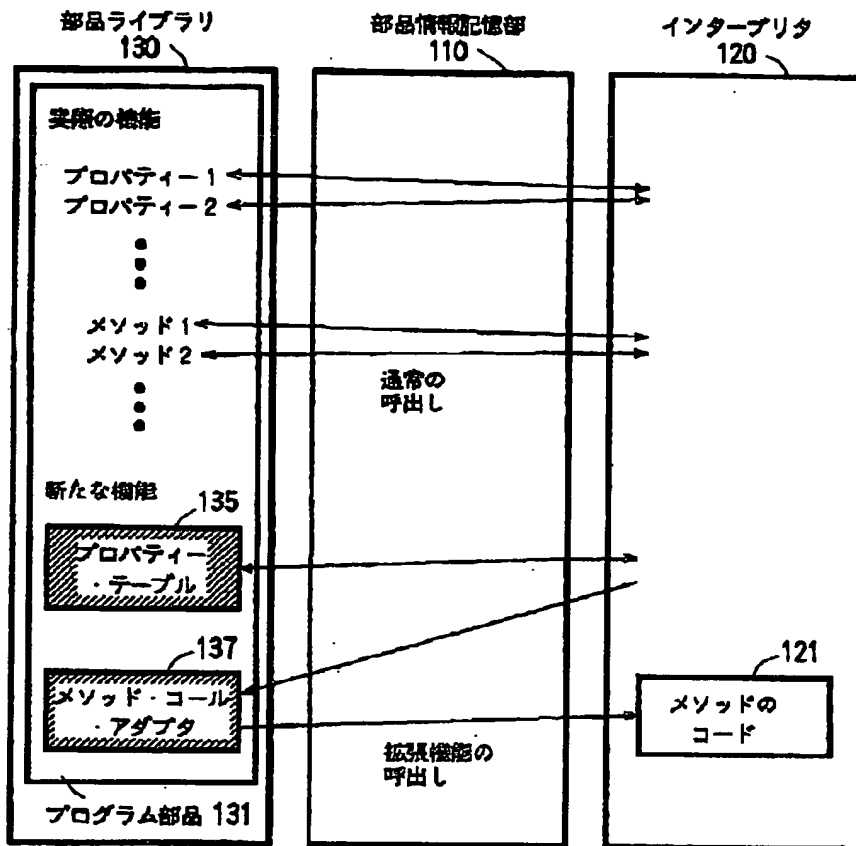


【図9】

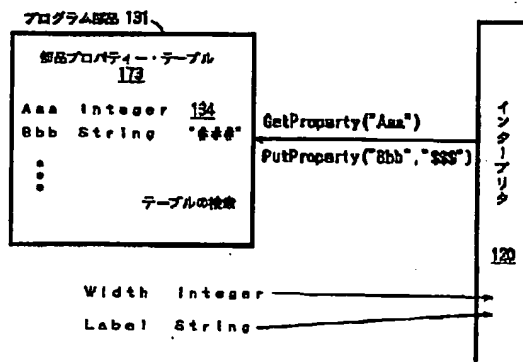




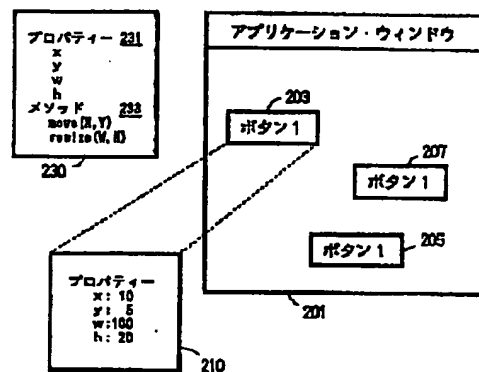
【図10】



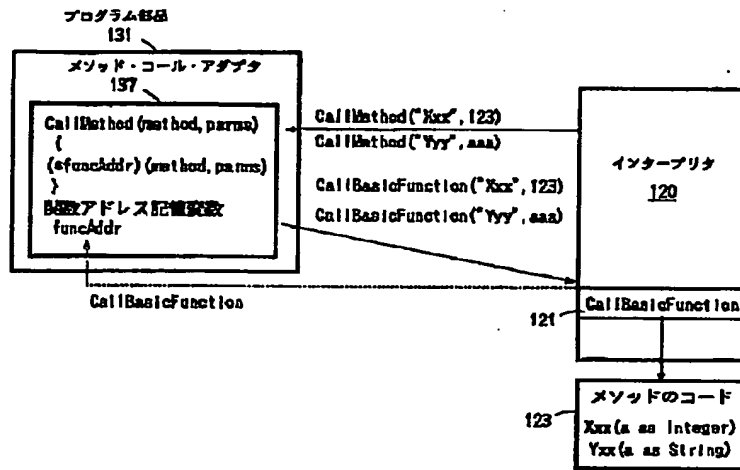
【図11】



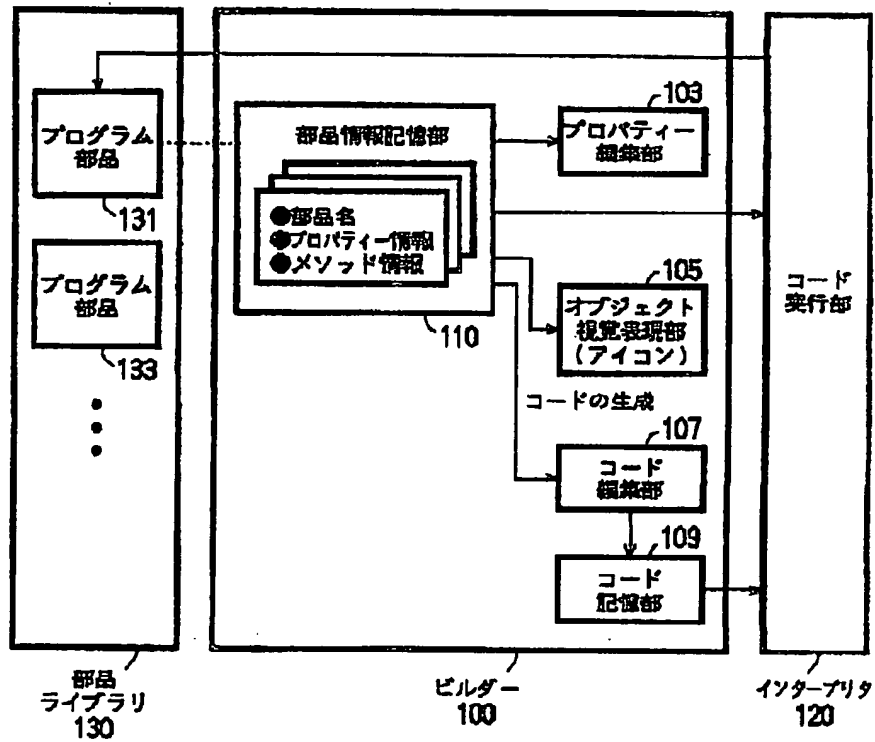
【図13】



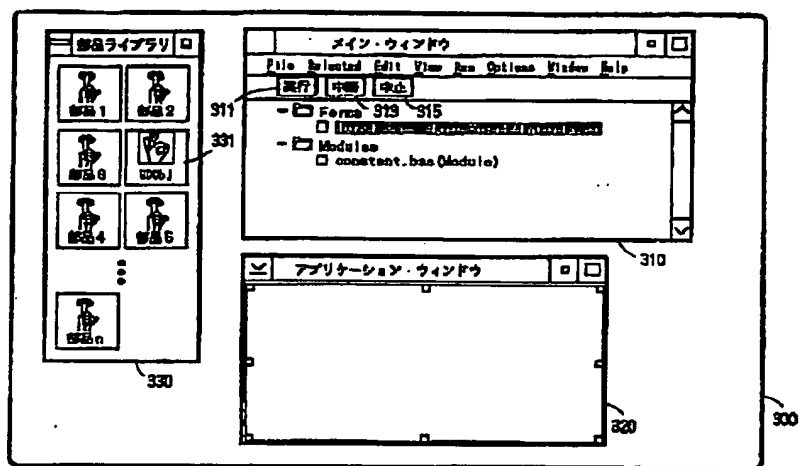
【図12】



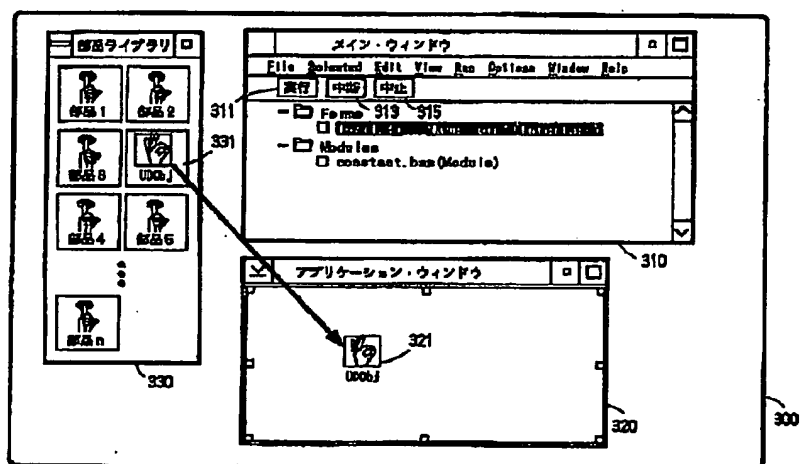
【図14】



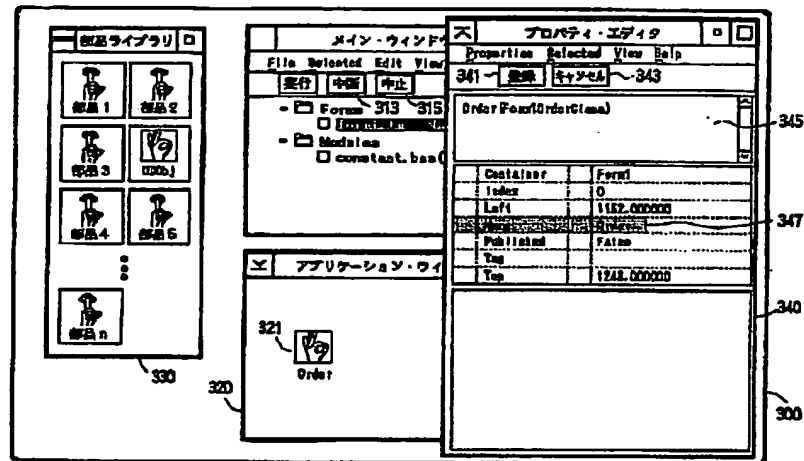
【図15】



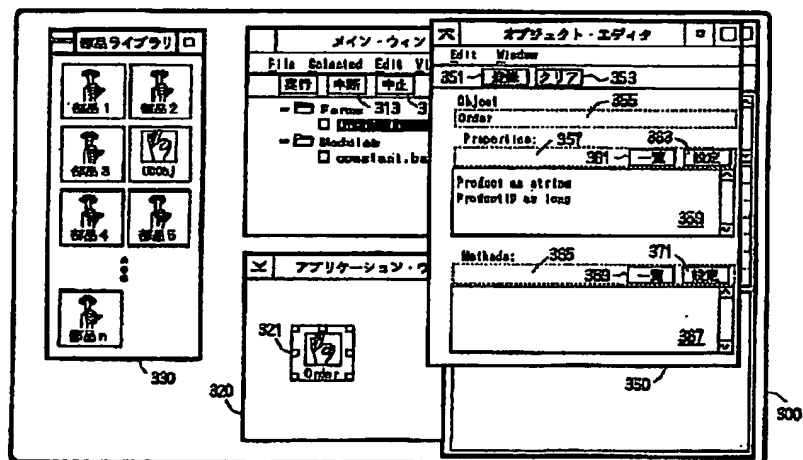
【図16】



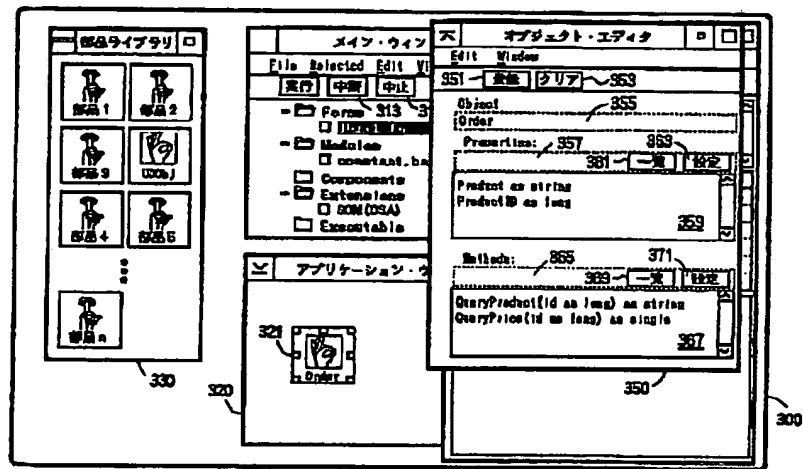
【図17】



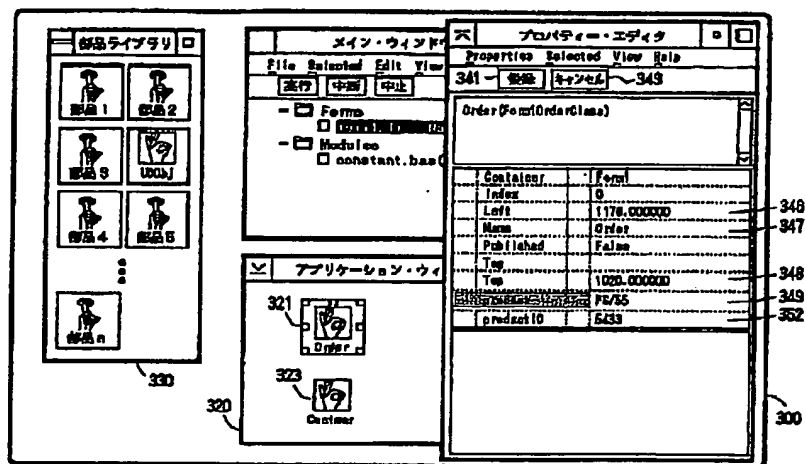
【図18】



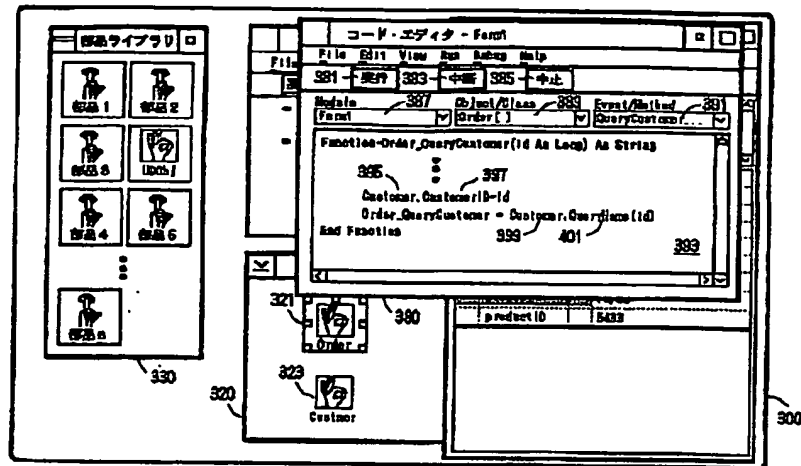
【図19】



【図20】



【図21】



フロントページの続き

(72)発明者 中村 祐一  
神奈川県大和市下鶴間1623番地14 日本ア  
イ・ピー・エム株式会社 大和事業所内

(72)発明者 田胡 和哉  
神奈川県大和市下鶴間1623番地14 日本ア  
イ・ピー・エム株式会社 大和事業所内